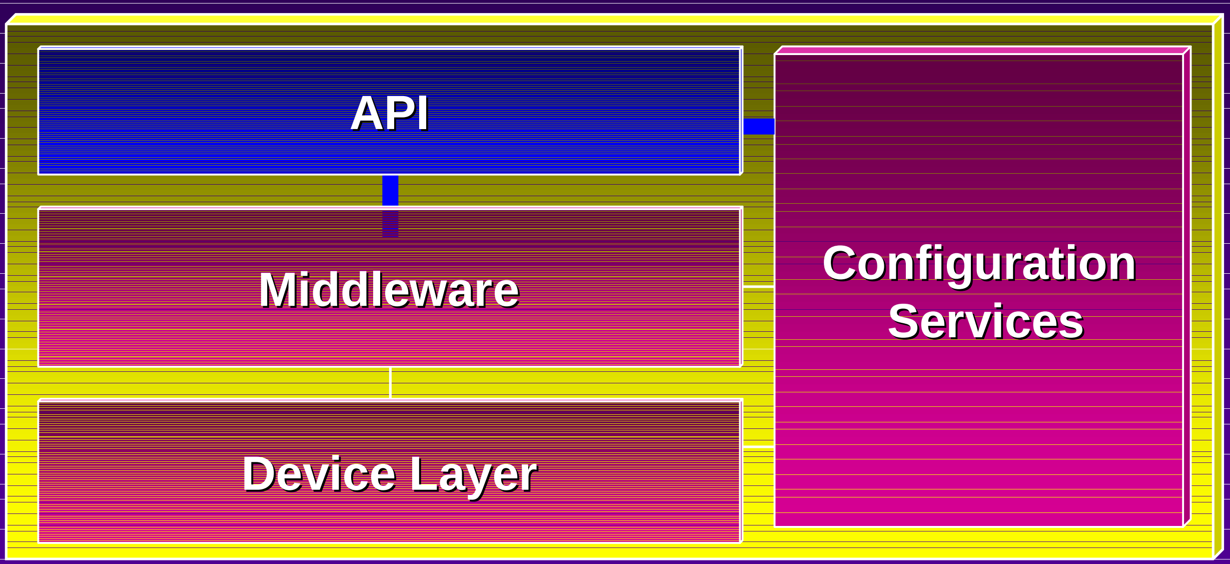
The Accelerator Device Model and the Java API

Franck Di Maio



The Common PS/SL API defines

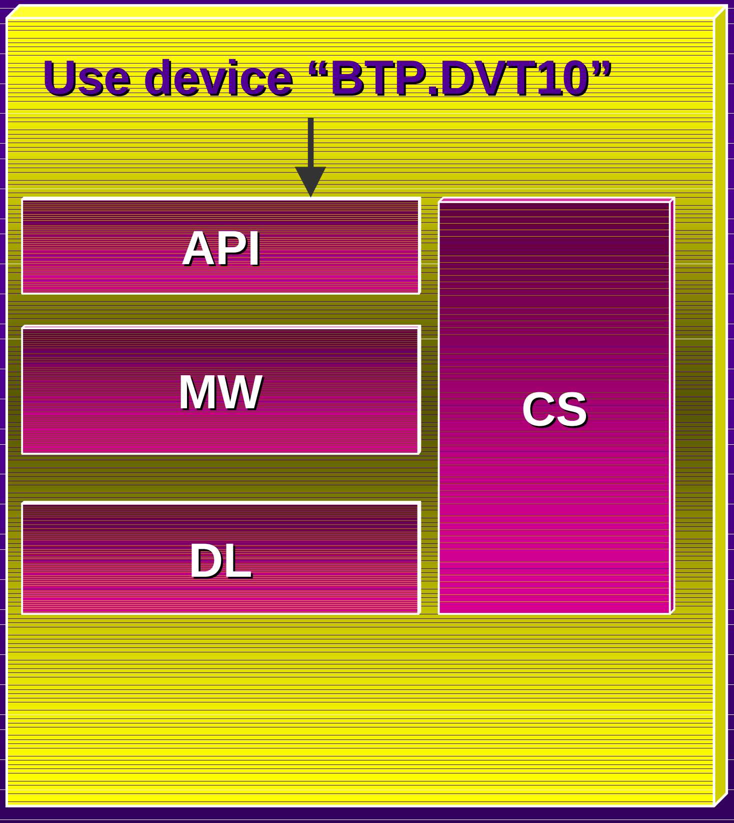
- the **structure of the objects**,
- the **services** that are available

for programs that control accelerators.

- The Device Model
- The I/O services
- Java Examples

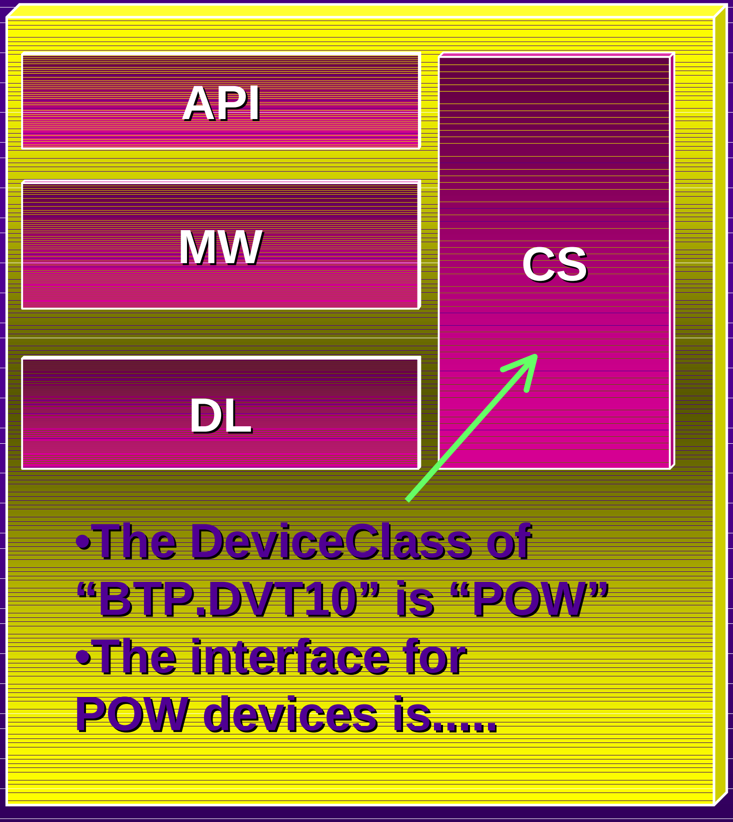
The Device Model (1)

- The system consists of named devices
  - a power-supply
  - a beam monitor
  - the PSB ring
  - etc.



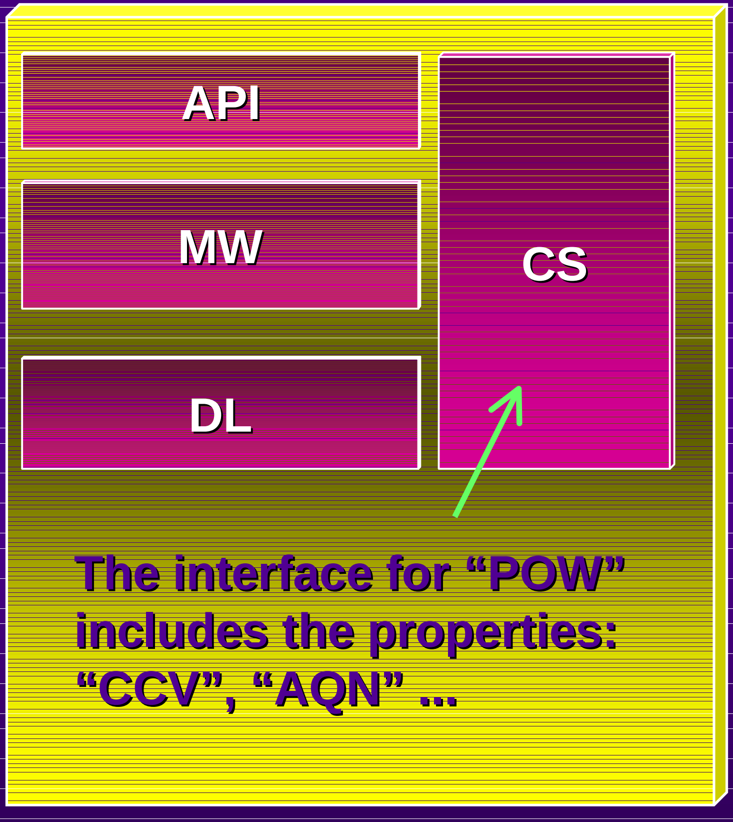
The Device Model (2)

- The devices are organized into device-classes
- The I/O interface of a device is described by its device-class


API
MW
DL
CS
•The DeviceClass of "BTP.DVT10" is "POW"
•The interface for POW devices is.....

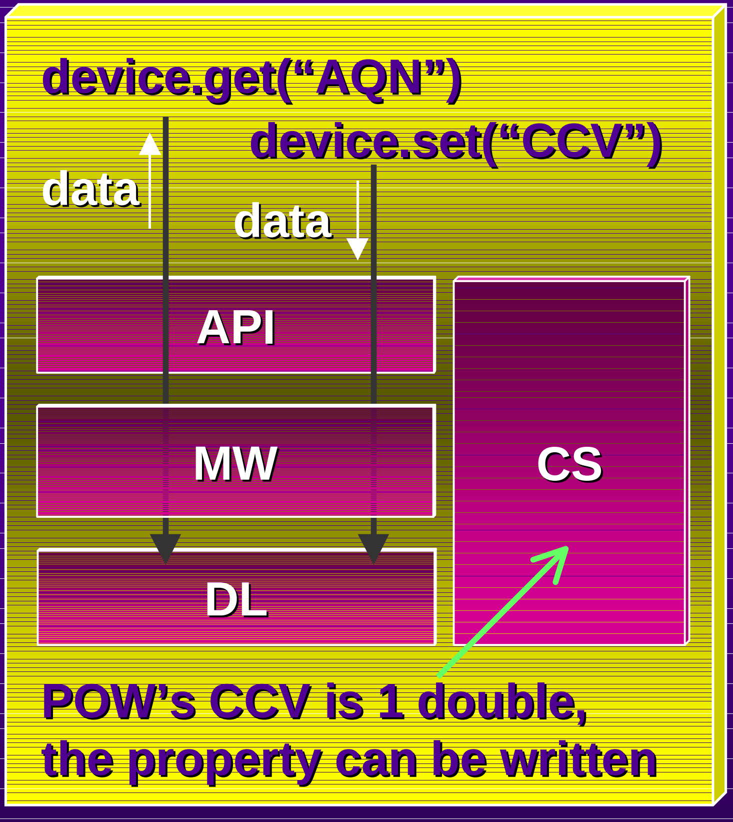
The Device Model (3)

- Any device value that varies from device to device (instance data) is interfaced by a named property.
- The devices' interface includes get/set methods that operate on properties.


API
MW
DL
CS
The interface for "POW" includes the properties: "CCV", "AQN" ...

I/O: get/set property

- Get property returns a value
- Set property sends a value
- Value's type:
  - scalars, string
  - arrays of scalars and strings
- Both may return an error information

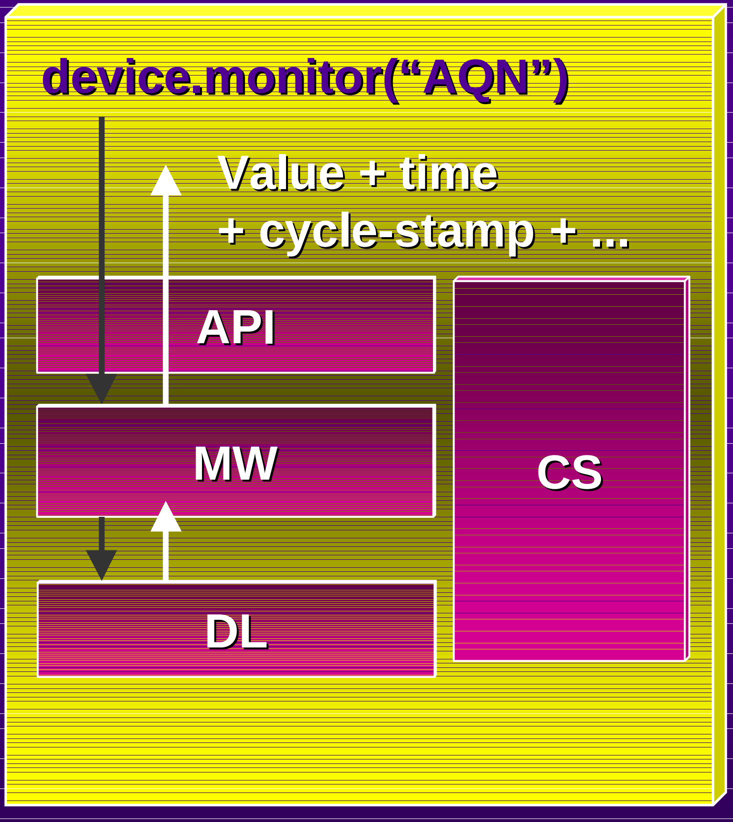


I/O: Data subscription

- A property that supports the get operation also supports the “subscribe / unsubscribe” operation
  - API method names: “monitorOn”, “monitorOff”
- Features:
  - The data is pushed by the data source.
  - The data acquisition is either periodic or triggered by a timing event.
  - The data can be delivered “on-change” only.

I/O: Acquisitions

- Acquisitions can be marked with:
  - a time-stamp (absolute, resolution: ms)
  - a cycle-stamp (unique for at least 24 hours)
  - an timing event
  - a cycle time (in ms since the start of the cycle)

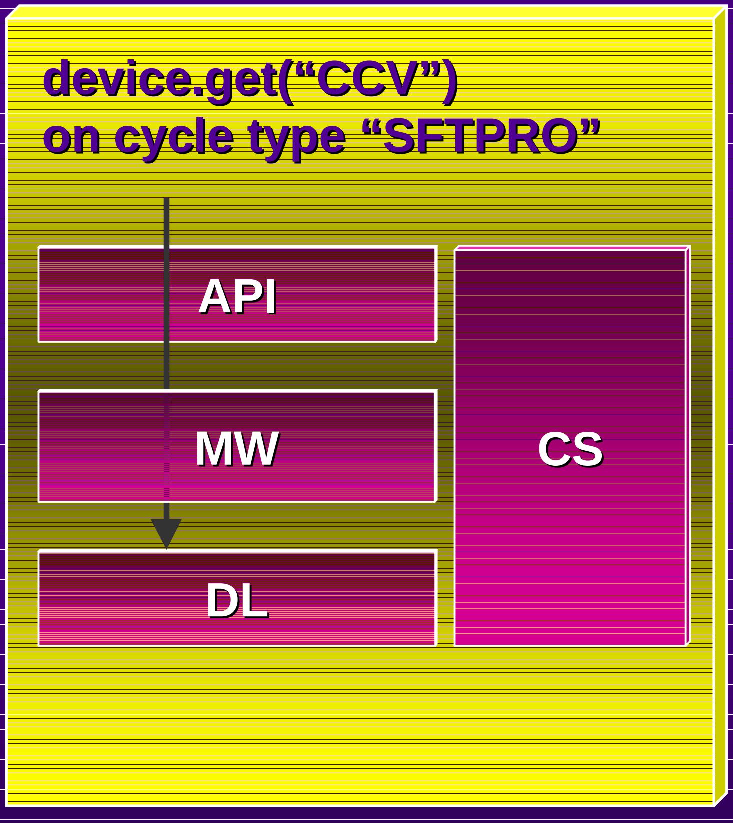


I/O: Errors

- Errors include at least:
  - A numeric code for the application.
  - An error message (string)
- Either an error or a value is returned, not both (get & monitorOn).

I/O:Timing System Conditions

- A device may be bound to a timing system that generates events and cycles.
- I/O operations may require:
  - a cycle type
  - a timing event



Ex.1: Simple get

```
// 1 - Create a device object
Device dev = new Device (“BTP.DVT10”);

// 2 - Read a property into a Data object
Data data = new Data();
DeviceError err = dev.get(“CCV”, data);

// 3 - Print results
if (err == null)
  System.out.println (data.getDataEntry(“value”).floatValue(),
                                        data.getDataEntry(“time”).doubleValue();
                                  data.getDataEntry(“cycleStamp”).intValue());
else
  System.err.println (err.getMessage());
```

Ex. 2: Monitor

```
// 1 - Create a device object
Device dev = new Device ("BTP.DVT10");

// 2 - Activate monitor with a listener
MyListener listener = new Listener();
dev.monitorOn("AQN", listener);

// Implementation of a listener
MyListener implements DeviceListener {
  void deviceChanged (DeviceEvent event) {
    DeviceError err = event.getError();
    if (err == null) {
      Data data = event.getData();

         ...
```

Conclusion

- A definition of the objects
  - Device, DeviceProperty, DeviceClass...
- A specification of the I/O services
  - operations, I/O parameters, transmitted data...
- Implementations of the Java API using the existing communication facilities

<u>Next</u>: a new communication architecture